tanarurkerem.hu
informatikai oktatási portál

Penceo
WE THINK DESIGN

Simply Reliable
NewPush

Integral Vision

5net
INTERAKTÍV TERVEZÉS ÉS FEJLESZTÉS

Bob nagybácsi és a programozás

Tasnádi Zsolt

Pitech+PLUS

2014. november 16.

Rólam

- vezető fejlesztő - Pitech+PLUS
- 2009-től foglalkozom webprogramozással
- Drupal, Symfony, Magento, stb.
- Drupal profile: `https://www.drupal.org/u/skipyt`
- IRC nick: **skipyT**
- Twitter: **@ztasnadi**

Bob nagybácsi

Figure : Bob Cecil Martin

a SOLID programozási elvek megalkotója

S.O.L.I.D.

- kedves kis betűszó: amelynek minden karaktere egy-egy alapelvet jelképez
- egy betűszó, ami betűszavakból áll

SRP - **S**ingle Responsibility Principle
OCP - **O**pen/Closed Principle
LSP - **L**iskov Substitution Principle
ISP - **I**nterface Segregation Principle
DIP - **D**ependency Inversion Principle

Single Responsibility Principle

- egy osztálynak/metódusnak egy, és pontosan egy oka lehet arra, hogy megváltozzon
- egy osztálynak/metódusnak egyetlen feladatot kell ellátnia

Open/Closed Principle

- a Nyitott/Zárt elv
- osztályaink legyenek nyitottak kiterjesztésre, de zártak módosításra

Liskov Substitution Principle

- *"Ha S altípusa T-nek, akkor minden olyan helyen ahol T-t felhasználjuk S-t is minden gond nélkül behelyettesíthetjük anélkül, hogy a programrész tulajdonságai megváltoznának."*
- magyarul: **Ha S osztály T osztály leszármazottja, akkor S szabadon behelyettesíthető minden olyan helyre (paraméter, változó, stb. . .), ahol T típust várunk.**
- téglalap-négyzet probléma

Interface Segregation Principle

- az alkalmazás *"alkatrészei"* minél lazábban kapcsolódjanak egymáshoz
- Man osztály implementálja a WalkerInterface-t (walk metódus) és a TalkerInterfacet (talk metódus), ahelyett, hogy a Personinterface-t implementálja (walk, talk metódus)
- Dog osztály implementálhatja a WalkerInterface-t

Dependency Inversion Principle

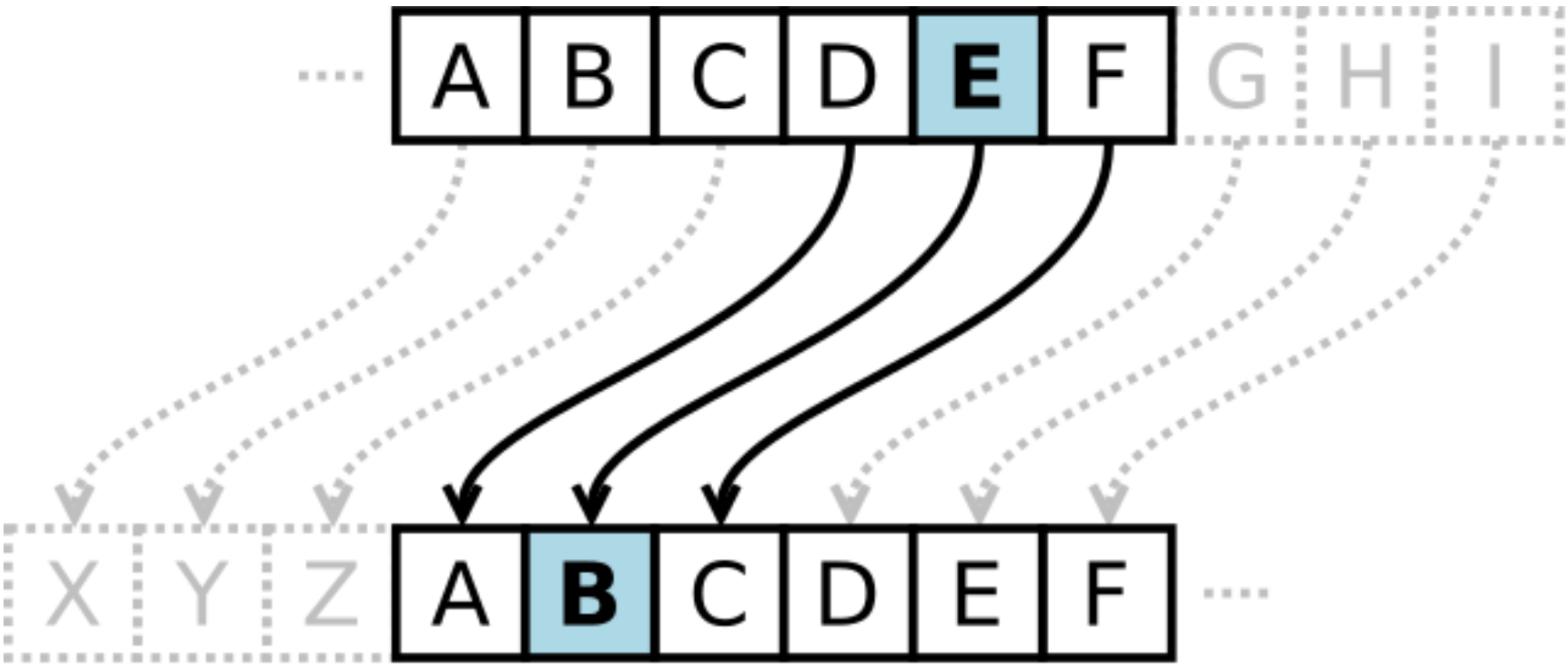
- A magasabb szinten lévő programegységek nem függhetnek az alacsonyabb szintűektől, ehelyett mindkettőnek az absztrakciótól kell függenie.
- Az absztrakció nem függ a részletektől, a részletek függnek az absztrakciótól.
- egyszerűen: Dependency Injection, így már hallottatok róla...

Feladat

Az NSA titkosított üzeneteket küld az FBI-nak egy RSS feeden keresztül. A mi feladatunk az, hogy ellemezzük az üzenetet és a megfejtése után tároljuk egy adatbázisban.

Titkosítás

Az NSA a Caesar-féle rejtjelzést használja.



Hova tároljuk

Drupal 8?
Tároljuk az időpontot mikor megkaptuk az üzenetet és a megfejtését.
Megoldás: **KeyValue**

Első probálkozás

```
class RSSMessagesParser {
  public function execute() {
    $nsaKeyValueStore = \Drupal::keyValue('nsa');
    $messages = $this->fetchMessages();
    foreach ($messages as $message) {
      $nsaKeyValueStore->set(time(),
        $this->decrypt($message));
    }
  }
```

Üzenetek letöltése

```
protected function fetchMessages() {
  return RSSParser::create(
    'http://secret.nsa.gov/rs'
      )->parse()->items();
}
```

Titkosítás megfejtése

```
    protected function decrypt($message) {
      $alphabet = range('a', 'z');

      $encrypted = preg_split("//",$msg, -1,
        PREG_SPLIT_NO_EMPTY);

      $decode['␣'] = '␣';
      foreach ($alphabet as $n => $v){
        $decode[$v] = $alphabet[(26 + ($n - 3)) % 26];
      }

      foreach ($encrypted as $v){
        $decrypt[] = $decode[$v];
      }

      return join('',$decrypt);
    }
}
```

Mi a baj ezzel?

- Mi lesz ha a URL megváltozik?
- Mi lesz ha változik a titkosítás?
- Mi lesz ha más helyről szeretnénk olvasni az üzeneteket?

Lássuk mit csinál az RSSMessagesParser

Letölti az üzeneteket
és
megfejti az üzeneteket
és
lementi az üzeneteket.

Következtetés

Az **és** meg a **vagy** szavak sértik az SRP-t. Az osztályunk túl sokat tud.

Osszuk szét külön osztályokra

```
class RSSMessagesParser {
  public function __construct() {
    $this->nsaCipher = new NSACipher();
    $this->rssMessages = new RSSMessages();
  }

  public function execute() {
    $messages = $this->rssMessages->fetch();
    $nsaKeyValueStore = \Drupal::keyValue('nsa');
    foreach ($messages as $message) {
      $nsaKeyValueStore->set(time(),
        $this->nsaCipher->decrypt($message));
    }
  }
}
```

SRP a metódusokba is

```
  public function execute() {
    $messages = $this->rssMessages->fetch();
    foreach ($messages as $message) {
      $this->parse($message);
    }
  }

  protected function parse($message) {
    $nsaKeyValueStore = \Drupal::keyValue('nsa');

      $nsaKeyValueStore->set(time(),
        $this->nsaCipher->decrypt($message));
    }
  }
}
```

NSACipher osztály

```
class NSACipher {
  public function decrypt($message) {
    //
  }
}
```

RSSParser osztály

```
class RSSMessages {
  public function fetch() {
    //
  }
}
```

Hogyan tovább

Az RSS url és a titkosítási lépés gyakran fog változni, próbáljuk meg alkalmazni az OPEN/CLOSE PRINCIPLE-t.

```
class NSACipher {
  public function __construct($step = 3) {
    $this->step = $step;
  }
  public function decrypt($message) {
    //
  }
}
```

tehát...

- Az osztály most nyitott a kiterjesztésekre és újra felhasználható.
- Zárt a módosításokra.
- az NSACipher osztály most egy általános Ceaser megfejtő és átnevezhetjük.

S akkor az RSSParser osztály

```
class RSSMessages {
  public function __construct($url =
    'http://secret.nsa.gov/rss') {
    $this->url = $url;
  }
  public function fetch() {
    //
  }
}
```

Konfiguráció hova?

De mi 8-as Drupal-t használunk, tehát használhatjuk a State service-t:

```
class RSSMessages
  implements ContainerInjectionInterface {
  public function __construct($url) {
    $this->url = $url;
  }

  public static function
    create(ContainerInjectionInterface $container) {
      return new
        static($container->get('state')
          ->get('RSS_url', 'http://secret.nsa.gov/rss
  }

  public function fetch() {
    //
  }
}
```

Dependency inversion

Drupal 8: használhatunk service-ket és DI-t.

```
services:
  nsa.messages_parser:
    class: Drupal\nsa\RSSMessagesParser
    arguments: ['@keyvalue', '@cipher', '@fetcher']
  cipher:
    class: Drupal\nsa\CeaserCipher
  fetcher:
    class: Drupal\nsa\RSSParser
```

RSSMessagesParser DI után, service-ként

```
class RSSMessagesParser {
  public function __construct(
    KeyValueFactoryInterface $key_value_factory,
    FetcherInterface $fetcher, CipherInterface $ciphe
    $this->nsaMessages =
      $key_value_factory->get('nsa_messages');
    $this->fetcher = $fetcher;
    $this->cipher = $cipher;
  }

  public function execute() {
    //
  }

  public function parse($message) {
    //
  }
}
```

Az eredmény

- Kicserélhetjük, hogy hova szeretnénk lementeni az üzeneteket (MySQL, MongoDB, stb.).
- Most kaphatjuk az üzeneteket különböző helyekről: állomanyok, RSS, Twitter, stb.
- Használhatunk különböző titkosítási algoritmusokat: AES, DES...
- Ha egy dolog változik, egy helyen kell változtatni
- Unit teszteket írni roppant egyszerű.

Köszönöm

Kérdések?